

Digitaler Zwilling eines Savonius-Rotors mit einer Simulink geregelten Fluent CFD Simulation

Manuel Berger, MCI - The Entrepreneurial School, Österreich
Thomas Senfter, MCI - The Entrepreneurial School, Österreich
Andreas Ravazzolo-Mehrle, The MathWorks Inc., Germany
Martin Pillei, MCI - The Entrepreneurial School, Österreich

Abstract:

Im vorliegenden Beitrag wird zu diesem Zwecke am Beispiel eines drehzahlgeregelten Savonius Rotors das kommerzielle Finite Volumen Programm ANSYS-FLUENT mit dem Softwarepaket Simulink gekoppelt. Dies ermöglicht einerseits eine akkurate Simulation der Strömung durch die erprobten Turbulenzmodelle des Strömungssolvers und andererseits die Verwendung komplexer Regelalgorithmen welche in Simulink zur Verfügung stehen. Der Datenaustausch erfolgt über eine Schnittstelle für welchen ein eigenes Modul in MATLAB/Simulink geschrieben wurde. Die Simulation erfolgt in 2D und nutzt lokal ein rotierendes Referenzsystem mit sliding mesh, welche in die restliche Topologie eingebaut wird.

Die Ergebnisse zeigen, dass der digitale Zwilling eine Nachbildung des realen Systems ohne ausufernden Programmieraufwand ermöglicht und eine Grundlage für die Optimierung von Steuerungsstrategien bietet. Der Ansatz demonstriert das Potenzial digitaler Zwillinge für die Entwicklung und virtuelle Inbetriebnahme von Systemen in der Strömungsmechanik.

Keywords: Digitaler Zwilling, Savonius Rotor, Numerische Strömungsmechanik, Discrete Regelung

1. EINLEITUNG

Die Entwicklung und Optimierung von Produkten in der Automatisierungstechnik erfordert zunehmend den Einsatz von digitalen Zwillingen und virtueller Inbetriebnahme. Besonders bei der Interaktion mehrerer physikalischer Domänen, wie z.B. bei der Rotation eines Rotors, ist die präzise Simulation von in diesem Fall Strömungsdynamik, Mechanik und Regelungstechnik entscheidend. Herkömmliche Ansätze, die auf physischen Prototypen basieren, sind kostenintensiv und zeitaufwendig. Die Integration von Computational Fluid Dynamics (CFD) mit Software zur Regelungssimulation bietet eine innovative Möglichkeit, die Produktentwicklung zu beschleunigen und die Effizienz zu steigern.

Die Herausforderung besteht darin, eine präzise und effiziente Kopplung zwischen der CFD und der Regelungssimulation zu schaffen, um einen digitalen Zwilling zu realisieren. Dabei müssen die transienten Strömungsdaten in Echtzeit verarbeitet und im Regelkreis integriert werden. Die bestehende Softwarepakete, wie FLUENT und Simulink, bieten zwar leistungsstarke Werkzeuge, jedoch ist die direkte Verbindung zwischen beiden Programmen technisch anspruchsvoll und erfordert eine maßgeschneiderte Lösung.

Das vorliegende Projekt nutzt die Fähigkeit von FLUENT, als Server zu agieren, um eine Verbindung mit Simulink herzustellen. FLUENT wird zur Berechnung der Strömungsgrößen eingesetzt, während

Simulink die Bewegungsgleichung löst und den Regeleingriff bestimmt. Die Kopplung erfolgt über Level-2-S-Funktionen, die eine bidirektionale Kommunikation zwischen den Programmen ermöglichen. Die Implementierung umfasst folgende Schritte:

- Erstellung des digitalen Zwillings: 2D Diskretisierung des Rotors und seiner Umgebung inklusive physikalischer Eigenschaften des Fluids.
- Integration der Regelung: Die Regelung des Rotors wird in Simulink implementiert, wobei die aggregierten Strömungsdaten aus FLUENT als Eingangsgrößen dienen.
- Optimierung der Schnittstelle: Die Verbindung zwischen FLUENT und Simulink wird durch die Nutzung von S-Funktionen und einer serverbasierten Architektur realisiert, die eine effiziente Datenübertragung ermöglicht.
- Simulation und Validierung: Der digitale Zwilling wird in verschiedenen Szenarien getestet, um die Genauigkeit der Simulation und die Leistungsfähigkeit der Regelung zu validieren.

2. METHODEN

Im Rahmen dieses Projekts wird eine Kopplung der Softwarepakete MATLAB/Simulink und ANSYS FLUENT zur Lösung gekoppelter Differentialgleichungen sowie zur Realisierung geschlossener Regelkreise umgesetzt. FLUENT wird zur numerischen Berechnung der zweidimensionalen Strömungssimulation eingesetzt, während Simulink die regelungstechnische Anbindung übernimmt. Zur Abbildung der Geometrie einschließlich aller relevanten Randbedingungen wird in FLUENT eine sogenannte Case-Datei (Dateiendung .cas) erstellt. Diese enthält das diskretisierte Rechengitter der Geometrie, die Definition der Randbedingungen wie Ein- und Auslässe sowie Wandbedingungen, ebenso wie die vorgegebenen Strömungseigenschaften. Nach der Erstellung kann die Case-Datei erneut geladen werden und dient als Ausgangsbasis für die Durchführung der Simulation.

2.1. FLUENT als Server

Zur Realisierung der Schnittstelle zwischen MATLAB und ANSYS FLUENT wird die Strömungssimulationssoftware im sogenannten Server-Modus gestartet. In diesem Betriebsmodus kann FLUENT vollständig über MATLAB-Befehle angesteuert und kontrolliert werden. Als Grundlage für die Implementierung dient eine bereits vorhandene Schnittstelle, die von einem Mitarbeiter der Firma ANSYS entwickelt wurde (Munteanu, 2014). Sofern innerhalb des verwendeten Netzwerks keine Einschränkungen durch Firewalls bestehen, ist die Einrichtung einer zusätzlichen Verbindung, beispielsweise über SSH, nicht erforderlich.

2.2. Verbindung mit Simulink

Im ersten Schritt wird die Windows-Konsole (cmd) mit Administratorrechten gestartet und in das gewünschte Arbeitsverzeichnis von ANSYS FLUENT gewechselt. Dabei ist sicherzustellen, dass der User über FLUENT über Schreibrechte im ausgewählten Verzeichnis verfügt. Anschließend wird FLUENT mit dem Befehl *fluent 3ddp -aas* gestartet. Diese Optionen legen fest, dass die Simulation dreidimensional erfolgt, der Solver mit doppelter Genauigkeit (Double Precision) arbeitet und FLUENT im sogenannten Server-Modus ausgeführt wird. Letzterer ermöglicht den externen Zugriff durch andere Programme.

Beim Start im Server-Modus wird automatisch eine Datei (aaS_FluentId.txt) erzeugt, die die für den Netzwerk-Client notwendigen Informationen zur eindeutigen Verbindungsherstellung enthält. Im

nächsten Schritt wird die zuvor erstellte Case-Datei mithilfe des Befehls `rc` (read case) geladen. Die anschließende Lösung der Strömungssimulation kann entweder über den Konsolenbefehl `solve it` oder über die grafische Benutzeroberfläche erfolgen. Zur Herstellung der Verbindung zwischen MATLAB und dem laufenden FLUENT-Server sind lediglich wenige Codezeilen erforderlich (vgl. Code 1). Dabei ist darauf zu achten, dass sich die Schlüsseldatei im aktuellen Arbeitsverzeichnis von MATLAB befindet. Wird FLUENT direkt aus MATLAB heraus gestartet, wird die entsprechende Schlüsseldatei automatisch im jeweiligen Arbeitsverzeichnis abgelegt.

```

fluent = actxserver('ANSYS.CoFluentUnit.1');
tui=fluent.getSchemeControllerInstance();
fid=fopen('aaS_FluentId.txt','r')
fluentkey=fscanf(fid,'%s')
fclose(fid)
fluent.ConnectToServer (fluentkey)

```

Code 1: Erstellung der Verbindung zwischen MATLAB und FLUENT.

Die ersten beiden Codezeilen dienen der Initialisierung der Serververbindung, wobei die Abkürzung *tui* (Text User Interface) die Konsoleneingaben von FLUENT repräsentiert. Anschließend wird der eindeutige Identifikationscode aus der Schlüsseldatei eingelesen und in einer neuen MATLAB-Variable gespeichert. Durch diese Schritte wird die Verbindung zwischen MATLAB/Simulink und ANSYS FLUENT hergestellt. Ein wesentlicher, im referenzierten Tutorial nicht explizit beschriebener Schritt besteht darin, den FLUENT-Server innerhalb der Windows-Umgebung zu registrieren. Dies ist Voraussetzung für eine stabile Kommunikation zwischen den Programmen. Die Registrierung erfolgt am einfachsten durch das Starten der Windows-Konsole (`cmd`) mit Administratorrechten und die anschließende Ausführung des entsprechenden Befehls (siehe Code 2). Dabei ist zu beachten, dass sich der Pfad zur FLUENT-Installation je nach Arbeitsplatz und verwendeter Programmversion unterscheiden kann, da neuere Versionen abweichende Verzeichnis-bezeichnungen verwenden (z. B. `v145`, `v150`, `fluent14.5.7`, `fluent15.0.0`).

```

regsvr32 "C:/Program Files/ANSYSInc/v145/fluent/fluent14.5.7/addons/corba/win64
/COMCoFluentUnit.dll

```

Code 2: Registrierung FLUENT Server in Windows.

Die beschriebene Methode dient der Herstellung der Serververbindung zwischen MATLAB/Simulink und ANSYS FLUENT. Um eine eindeutige und fehlerfreie Verbindung sicherzustellen, empfiehlt es sich, beim Beenden des Programms die zuvor erzeugte Schlüsseldatei zu löschen. Dadurch wird verhindert, dass bei einem erneuten Verbindungsaufbau auf eine abgelaufene FLUENT-Sitzung zugegriffen wird.

Nachher startet der Berechnungsprozess:

Innerhalb einer MATLAB-m-Datei wird eine Schleife ausgeführt, in der jeweils ein Iterationsschritt der FLUENT-Simulation berechnet wird, bis die definierte Anzahl an Schleifendurchläufen erreicht ist. Voraussetzung hierfür ist das Laden einer vorbereiteten Case-Datei, die sowohl das Rechengitter als auch die definierten Randbedingungen enthält. Zusätzlich werden zwei Parameter zur Ein- und Ausgabe des Systems festgelegt, deren Rückmeldungen in der MATLAB-Konsole ausgegeben werden. Treten dabei Fehler auf, werden entsprechende Fehlermeldungen angezeigt. Das Laden einer Daten-Datei ist optional. Wird als Eingabeparameter ein leerer String übergeben, führt FLUENT eine hybride Initialisierung durch und startet die Simulation ohne bekannten Startwerten. Soll hingegen eine Daten-

Datei verwendet werden, wird zunächst die Case-Datei geladen, anschließend und danach die Daten-Datei importiert. Dieser Ablauf stellt die Initialisierung dar, die FLUENT für den Start der Simulation vorbereitet. Wird beispielsweise eine Zeitschrittweite von 0,1 s bei einer maximalen Iterationsanzahl von 50 gewählt, erstreckt sich die Simulation über einen Zeitraum von 0 bis 5 s. Der Parameter maximale Anzahl an Iterationen pro Zeitschritt beschreibt einen internen FLUENT-Wert, der vorgibt, wie viele Iterationen zur Konvergenz innerhalb eines Zeitschritts zulässig sind. Nicht konvergierende Zeitschritte werden ebenfalls in der MATLAB-Konsole ausgegeben. Nach Abschluss der Simulation wird eine Pixelgrafik des Residuenverlaufs im Arbeitsverzeichnis gespeichert. Code 3 zeigt beispielhaft die Ausgabe der MATLAB-Konsole.

```
Case-file successfully imported
No dat file desired for the simulation
Calculate time step 1
ATTENTION: solution is not converged in time step 1
Calculate time step 2
Calculate time step 3
Calculate time step 4
Calculate time step 5
Code 3: MATLAB Konsolenausgabe bei Berechnung von MATLAB mit FLUENT.
```

Zu Beginn der Simulation wird eine Case-Datei geladen; auf die Verwendung einer Daten-Datei wird verzichtet. Anschließend startet FLUENT den iterativen Lösungsprozess. Dabei zeigt sich, dass im ersten Zeitschritt keine Konvergenz der Residuen innerhalb der maximal vorgegebenen Iterationsanzahl erreicht wird, während alle nachfolgenden Zeitschritte eine stabile Konvergenz aufweisen.

Parallel zur Strömungssimulation wird ein PI-Regler implementiert, der so ausgelegt ist, dass keine bleibende Regelabweichung im stationären Zustand auftritt. Hierzu wird der momentane Regelabweichungswert mit einem konstanten Proportionalfaktor multipliziert und zusätzlich die zeitliche Summe der Regelabweichung zur Berechnung der Stellgröße herangezogen.

```
output_system(1)=0;
sum_error(1)=0;

for i = 1:number_timesteps
    control_error(i)=desired_value(i)-output_system(i);
    controller_output=control_error(i)*P_value+I_value*sum_error(i)*str2num(time_step_size);
    fluent.setInputParameterValueByName(input,controller_output);
    tui.DoMenuCommand(['/solve/dual-timeiterate 1 ', iterationpertimestep]);
    output_system(i+1)= fluent.getOutputParameterValueByName(output);
end
```

Code 3: PI Regler Realisierung in MATLAB zur FLUENT Regelung.

2.1. Simulink Erweiterung

Die Möglichkeit, weiterführende Regelungsstrategien zu realisieren, motiviert die Umsetzung des geschlossenen Regelkreises mit einem virtuellen FLUENT-Modell innerhalb der grafischen Programmierumgebung Simulink. Hierfür stellt MathWorks die Möglichkeit bereit, MATLAB-Code sowie weitere Programmiersprachen wie C, C++ oder Fortran über benutzerdefinierte S-Funktionen einzubinden. Der Begriff S-Funktion (System Function) bezeichnet ein leistungsfähiges Erweiterungswerkzeug zur Implementierung eigener Systemmodelle und Schnittstellen in Simulink. Grundsätzlich wird zwischen Level-1- und Level-2-S-Funktionen unterschieden. Erstere stellen eine frühere Entwicklungsstufe dar und bieten im Vergleich zu den moderneren Level-2-S-Funktionen einen geringeren Funktionsumfang. Beide Varianten können aus programmietechnischer Sicht als Programmierschnittstellen (APIs) verstanden werden. Da Level-2-S-Funktionen insbesondere für komplexe Programmieraufgaben empfohlen werden, kommen sie in diesem Projekt zum Einsatz. Zum Verständnis der Möglichkeiten dieses Ansatzes ist die Betrachtung der verschiedenen Simulationsphasen erforderlich, die in Abbildung 1 dargestellt sind.

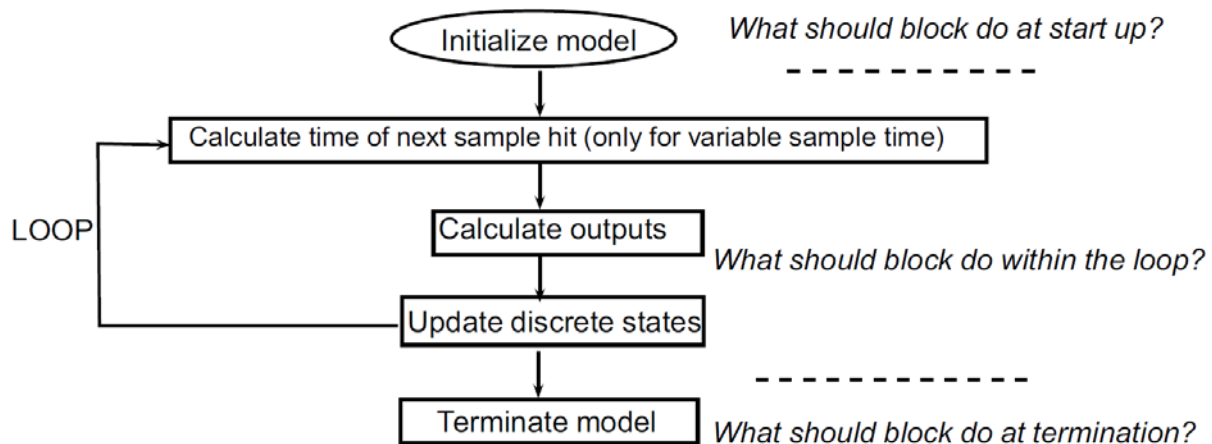


Abbildung 1: Status einer Level-2 s Funktion in Simulink. Genommen aus (Berger, 2014)

Der Programmcode gliedert sich grundsätzlich in drei Phasen: die Initialisierung beim Start des Simulink-Modells, die eigentliche Schleifenberechnung sowie die Terminierung. Diese Struktur stellt einen wesentlichen Unterschied zur Verwendung eines klassischen MATLAB-m-Files in Simulink dar, da dort eine derartige Phasentrennung nicht vorgesehen ist. Während der Initialisierungsphase werden sämtliche Schritte definiert, die in FLUENT vor Beginn der iterativen Strömungsberechnung erforderlich sind. Zu Beginn erfolgt das Laden der Case-Datei, gefolgt von der Übernahme aller notwendigen Definitionen und Einstellungen zur Vorbereitung einer lauffähigen Simulation. Die Simulink-Schleife arbeitet analog zu einer for-Schleife in MATLAB, wobei in jedem Schleifendurchlauf genau ein Zeitschritt berechnet wird. Hierbei verwendet der Solver die vorab definierte maximale Anzahl an Iterationen pro Zeitschritt. Ein weiterer Unterschied besteht darin, dass S-Funktionen nicht nur aus einem einzelnen Codeabschnitt bestehen, sondern eine Vielzahl an Strukturen und Konfigurationsparametern enthalten, um den Anforderungen komplexer Programmieraufgaben gerecht zu werden. Um den Zugriff auf die FLUENT-Serververbindung zu ermöglichen, muss der dargestellte Code 1 einmalig während der Initialisierung des Simulink-Modells ausgeführt werden. Zur Kopplung von FLUENT und Simulink ist darüber hinaus die Definition geeigneter Blockeigenschaften

erforderlich. Um eine modulare Struktur zu realisieren, werden die Funktionen nicht in einem einzelnen Block zusammengefasst, sondern auf insgesamt drei Simulink-Blöcke verteilt. Deren jeweilige Ein- und Ausgangsgrößen sind in Abbildung 2 dargestellt.

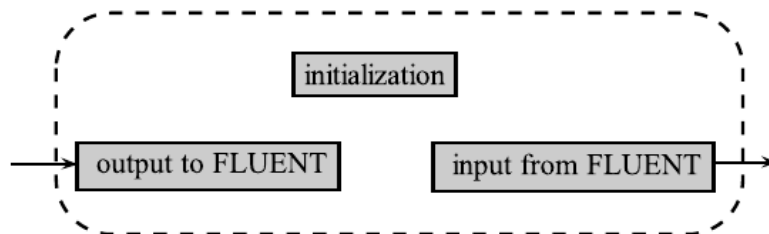


Abbildung 2: Struktur des Simulink Blocks. Genommen aus (Berger, 2014)

Es werden insgesamt drei S-functions in einem Block vorbereitet. Eine S-function dient der Initialisierung und umfasst sämtlichen Code, der erforderlich ist, um FLUENT für den Start der Simulation vorzubereiten. In diesem Schritt wird zunächst die Verbindung zum FLUENT-Server aufgebaut, anschließend werden die Case- und gegebenenfalls Daten-Datei geladen und der Simulationstyp auf transient gesetzt. Der in dieser S-Funktion implementierte Code wird ausschließlich einmal zu Beginn der Simulation ausgeführt. Die Platzierung dieses Codeabschnitts in der S-Function Start stellt sicher, dass die Initialisierung in der vorgesehenen Reihenfolge erfolgt; diese Struktur ist bereits im grundlegenden S-Funktions-Template enthalten. Zusätzlich werden die Ein- und Ausgänge dieser S-function auf null gesetzt. Dabei ist zu beachten, dass auch die Definitionen der S-function Eigenschaften innerhalb der Datei entsprechend auszukomentieren sind, da es andernfalls während des Kompilierungsprozesses zu Fehlern kommen kann. Um eine konsistente Simulationszeit in Simulink und FLUENT sicherzustellen, wird eine globale Variable, beispielsweise time, initial mit dem Wert null angelegt.

Der Block Output to FLUENT übernimmt die vom Regler berechnete Stellgröße und überträgt diese direkt an FLUENT. Der zugehörige Code wird während der Laufzeit der Simulation innerhalb der Schleife ausgeführt. Für diese Aufgabe stehen grundsätzlich mehrere Funktionsaufrufe zur Verfügung; zur Gewährleistung einer übersichtlichen und nachvollziehbaren Implementierung wird jedoch die Funktion Output (blocks) verwendet, da deren Werte in einem laufenden Simulink-Modell kontinuierlich aktualisiert werden. Darüber hinaus ist festzulegen, in welchem Block der jeweilige Iterationsschritt der Strömungssimulation durchgeführt wird. Grundsätzlich kann jede innerhalb der Schleife ausgeführte Funktion hierfür genutzt werden. Von besonderer Bedeutung ist dabei die Synchronisation der Simulationszeit in Simulink und FLUENT. Trotz der Einstellung eines festen Zeitschritts im Simulink-Solver kann die Verwendung kontinuierlicher Blöcke, wie beispielsweise eines PI-Reglers im s-Bereich, zu einer effektiven Veränderung der Schrittweite führen. In diesem Fall werden zusätzliche interne Subschritte eingeführt, um die kontinuierlichen Signale numerisch zu diskretisieren. Dieser Prozess verläuft nicht linear.

Infolgedessen muss während der Simulation die aktuelle Simulink-Zeit ausgelesen werden. Durch Subtraktion der Zeit des vorherigen Iterationsschritts lässt sich die tatsächlich verwendete Simulationszeitschrittweite bestimmen. Aufgrund der nichtlinearen Erzeugung interner Subschleifen und der daraus resultierenden variablen Zeitschritte ist es nicht möglich, die effektive Schrittweite zu

Beginn der Simulation eindeutig festzulegen. Ein wesentlicher Vorteil dieses Ansatzes besteht jedoch darin, dass die vollständige Funktionalität von Simulink genutzt werden kann, beispielsweise zur Realisierung komplexer geschlossener Regelkreise. In der Terminierungsphase wird FLUENT ordnungsgemäß beendet und die Schlüsseldatei gelöscht, um Konflikte bei nachfolgenden Simulationen zu vermeiden.

Der Block „Input from FLUENT“ übernimmt die aus FLUENT bereitgestellten Parameter und überträgt diese in das Simulink-Modell. Da der Iterationsschritt bereits in der „Output to FLUENT“-S-Funktion durchgeführt wird, ist an dieser Stelle keine weitere Berechnung erforderlich. Die drei beschriebenen Blöcke lassen sich zu einem Subsystem zusammenfassen, das in Abbildung 2 durch eine gestrichelte Linie dargestellt ist.

Zur Verbesserung der Wiederverwendbarkeit und Anwenderfreundlichkeit wird eine eigene Simulink-Bibliothek sowie eine benutzerdefinierte grafische Oberfläche erstellt. Dadurch kann die entwickelte Lösung mit minimalem Implementierungsaufwand weitergegeben werden, wodurch Fehler bei der praktischen Anwendung reduziert werden. Die Erstellung einer neuen Bibliothek unterscheidet sich nicht wesentlich von der eines herkömmlichen Simulink-Modells; auch die Dateiendung ist identisch. Ein Vorteil dieses Ansatzes besteht darin, dass Änderungen innerhalb der Bibliothek automatisch in allen Modellen übernommen werden, in denen der entsprechende Block verwendet wird. Zusätzlich kann der Zugriff auf die Bibliothek erleichtert werden, indem sie mithilfe der slblocks-Funktion in den Simulink Library Browser eingebunden wird. Bei einer gewählten Zeitschrittweite von 0,1 s und der Verwendung des Standard-Solvers ode3 (Bogacki–Shampine) werden beispielsweise zwei zusätzliche Subschritte innerhalb eines Simulationsschritts erzeugt.

2.2. MIMO Systeme

MIMO-Systeme (Multiple Input, Multiple Output) sind Systeme mit mehr als einem Ein- und Ausgang. Sämtliche definierten Ein- und Ausgangsparameter müssen im FLUENT-Case-File in einer festgelegten, chronologischen Reihenfolge angeordnet sein (I1, I2, I3, ... sowie O1, O2, O3, ...). Unabhängig von der Anzahl der Eingangsgrößen werden diese in FLUENT zu einem gemeinsamen Signalvektor zusammengefasst. Der erste Verarbeitungsschritt besteht daher darin, diesen Signalvektor in seine einzelnen Komponenten zu zerlegen, um die jeweiligen FLUENT-Eingangsparameter in jedem Iterationsschritt gezielt setzen zu können. Das entsprechende Vorgehen ist in Code 4 dargestellt.

```

Block_input = block.InputPort(1).Data;
for i=1:num_inputs
    fluent.setInputParameterValueByName(['I',num2str(i)], Block_input(i)); % Take the inputvalue and
    writes it to Fluent
end
  
```

Code 4: PI Regler Realisierung in MATLAB zur FLUENT Regelung.

2.3. Parallele FLUENT Simulation

Der Start von ANSYS FLUENT über die Windows-Kommandozeile oder direkt aus MATLAB erfolgt mit folgender Syntax: „fluent -t8 -pnmpi“. Die zusätzlichen Startparameter definieren die Nutzung von insgesamt acht Prozessorkernen unter Verwendung des standardmäßigen Job-Schedulers. Eine

zentrale Voraussetzung für die effiziente parallele Berechnung ist die Partitionierung des Rechengitters, da die Lösung anschließend auf mehrere Prozessoren verteilt berechnet wird. Hierfür besteht entweder die Möglichkeit, das Gitter automatisch partitionieren zu lassen oder eine manuelle Aufteilung vorzunehmen. Die automatische Partitionierung liefert in der Regel sehr gute Ergebnisse hinsichtlich der Lastverteilung der Prozessoren sowie der Minimierung gemeinsamer Randflächen zwischen den Teilgebieten. Dadurch erreichen die einzelnen Prozessoren ihre Berechnungen nahezu zeitgleich, wodurch unnötige Wartezeiten bei der Interprozesskommunikation vermieden werden.

3. ERGEBNISSE

3.1. Strömungsregelung

Das erste Anwendungsbeispiel befasst sich mit einem Massenstrom-Regelsystem unter Verwendung einer Drosselklappe, wie sie typischerweise in Verbrennungsmotoren eingesetzt wird. Dabei wird der Massenstrom durch die Stellung eines Ventils so angepasst, dass die gewünschte Leistungsabgabe erreicht wird. Um dieses System realitätsnah mithilfe einer FLUENT-Case-Datei abzubilden, sind mehrere vorbereitende Schritte erforderlich. Zunächst wird das Rechengitter der Geometrie erzeugt und sämtliche relevanten Randbedingungen definiert. Von zentraler Bedeutung ist hierbei, dass das verwendete Gitter deformierbar ausgelegt ist, da eine präzise Abbildung der Drosselklappenbewegung während der Simulation notwendig ist. Zur Umsetzung dieser Geometrieanpassungen wird eine benutzerdefinierte Funktion (User Defined Function, UDF) eingesetzt.

Abbildung 3 zeigt eine schematische Darstellung des konzeptionellen Aufbaus. Das System besteht aus einem Rohr, in dessen Mitte eine rotierend gelagerte Drosselklappe angeordnet ist, über die der Massenstrom $\dot{m}$ eingestellt wird. Unter der Annahme einer inkompressiblen Strömung tritt zu jedem Zeitpunkt dieselbe Fluidmenge in das Rechengebiet ein und verlässt dieses wieder. Zwischen Ein- und Auslassfläche besteht dabei eine Druckdifferenz. Diese kann beispielsweise durch einen Kolben, der Luft in einen Verbrennungsmotor ansaugt, oder durch einen Turbolader, der die Dichte an der Einlassfläche erhöht, verursacht werden. Bei einer horizontalen Ausrichtung der Drosselklappe wird der maximale Massenstrom durch das System erreicht. Aufgrund der rotatorischen Lagerung der Klappe kann der Durchfluss jedoch gezielt reduziert werden. Ziel ist die Realisierung eines Regelsystems zur Einstellung des Massenstroms $\dot{m}$, wobei der Systemeingang durch den Winkel α beschrieben wird, der die Stellung der Drosselklappe angibt. An den Wandflächen ist kein Ein- oder Austritt des Fluids aus dem Simulationsgebiet zulässig.

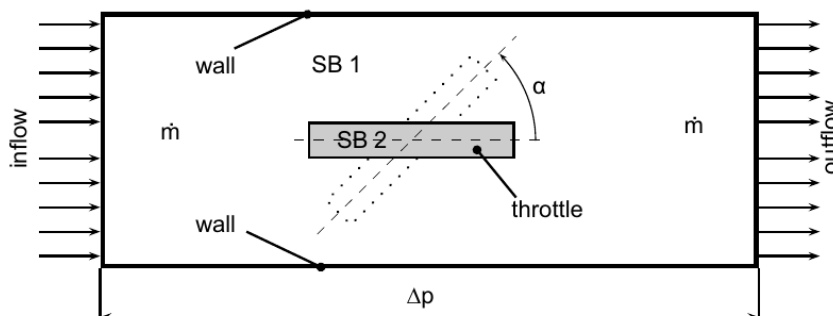


Abbildung 3: Problemstellung Drosselklappe, Modell. Genommen aus (Berger, 2014)

Um ein geschlossenes Regelungssystem zu realisieren, müssen zunächst die Schnittstellen zu Simulink geschaffen werden. Die Systemausgabe ist dabei relativ einfach, da sie lediglich den Massenstrom des Ausflusses darstellt (siehe Abb. 3). Als Systemeingang dient α , das die Orientierung der Drosselklappe relativ zur Horizontalachse definiert. Es wird also im Wesentlichen eine zusätzliche Verbindung zwischen Simulink und der C-Datei der UDF benötigt. Der erste Schritt besteht daher darin, in FLUENT eine neue, benutzerdefinierte Schema-Variable zu definieren: „(rp-var-define 'i1 0.2 'Real #f)“.

Abbildung 4 zeigt das erstellte Simulink-Modell. Auf der linken Seite befinden sich die Blöcke, die über das Subtract-Tool zusammengeführt werden. Grundsätzlich soll das Modell das dynamische Verhalten der Drosselklappe simulieren. Zusätzlich zum FLUENT-Modell wurde ein PT1-System eingefügt, um die Trägheit der rotierenden Drosselklappe abzubilden. Andererseits erhöht diese so eingeführte Zeitverzögerung die Stabilität, da schnelle Verformungen, die zu negativen Volumenzellen und einem sofortigen Abbruch der Simulation führen würden, nicht mehr möglich sind. Die zu regelnde Größe ist der Massenstrom $\dot{m}$. Da FLUENT einen negativen Wert erzeugt, weil das Fluid die Simulationsdomäne verlässt, wird dieser Wert mit der Simulink-Funktion in einen positiven Wert umgewandelt.

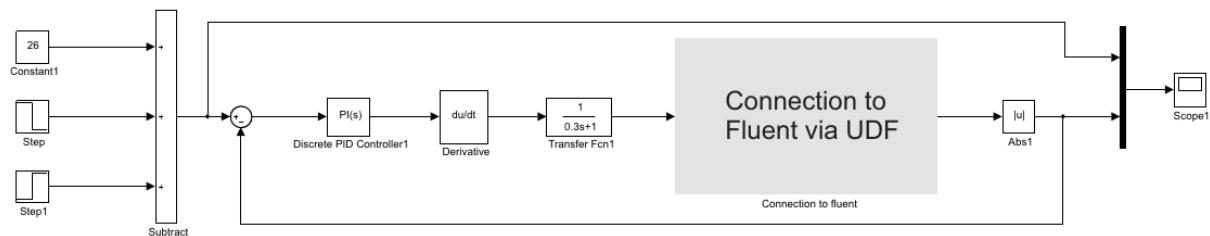


Abbildung 4: Strömungsregelung mit einer Drosselklappe, Model. Genommen aus (Berger, 2014)

Die UDF-Datei arbeitet mit Drehzahlen in der Einheit rad/s. Der gewünschte Systemeingang ist jedoch der Winkel α in rad. Diese Transformation wird durch einen zusätzlichen Derivationsblock zwischen der Systemübertragungsfunktion und dem Regler realisiert. Dieses Modell liefert das geschlossene Regelverhalten, das in Abbildung 5 dargestellt ist. Zu Beginn ist die Drosselklappe vollständig geöffnet, und der Regler versucht, nahezu den maximal möglichen Wert einzustellen. Dies erzeugt einen Massenstrom von etwa 26 kg/s. Anschließend wird praktisch der Effekt simuliert, der auftritt, wenn der Fahrer das Gaspedal löst und nur noch nahezu die minimale Gasmenge benötigt wird.

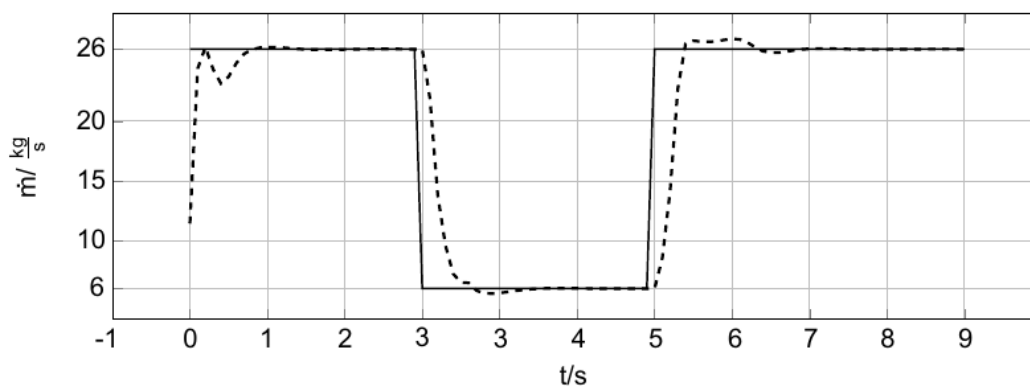


Abbildung 5: Strömungsregelung mit einer Drosselklappe, Ergebnisse. Genommen aus (Berger, 2014)

3.2. Savonius Rotor

Eine Savonius-Windturbine wird eingesetzt, um Windenergie in Rotationsbewegung umzuwandeln und dadurch Strom zu erzeugen. Sie gehört zu den einfachsten Turbinen; eine Zwei-Segel-Variante hat im Querschnitt die Form eines „S“. Die Bewegungsgleichung für die Rotation lautet $I\dot{\omega} + b\omega = M(t)$. Um diese Gleichung zu lösen, ist eine Transformation in eine Blockdarstellung notwendig, um sie in einem Simulink-Modell implementieren zu können. Der erste Schritt besteht darin, die Gleichung so umzustellen, dass die höchste Ableitung auf einer Seite der Gleichung steht. Sie kann dann wie folgt geschrieben werden $\dot{\omega} = \frac{M(t) - b\omega}{I}$. Die Bewegung des Rotors kann in FLUENT durch einen rotierenden Bezugskörper dargestellt werden. Für eine gute Systemdarstellung ist die richtige Struktur entscheidend. Abbildung 6 zeigt das verwendete Netz. Zunächst werden beide Bereiche im Ansys Design Modeler als Flächen erzeugt. Wichtig ist, dass beide zu einem neuen Bauteil zusammengeführt werden; andernfalls liegen die Knoten der beiden Netzbereiche im Schnittbereich nicht übereinander, und Informationen werden nicht korrekt übertragen. Der Rotor selbst wird nicht vernetzt, da sich in diesem Bereich keine Strömung befindet und keine strukturelle Analyse erforderlich ist. Zusätzlich werden benannte Flächen (Named Selections) erstellt. Die Rotorwand wird für den rotierenden Bezugskörper benötigt, die Außenwand dient als Randbedingung, an der keine Energie übertragen wird.

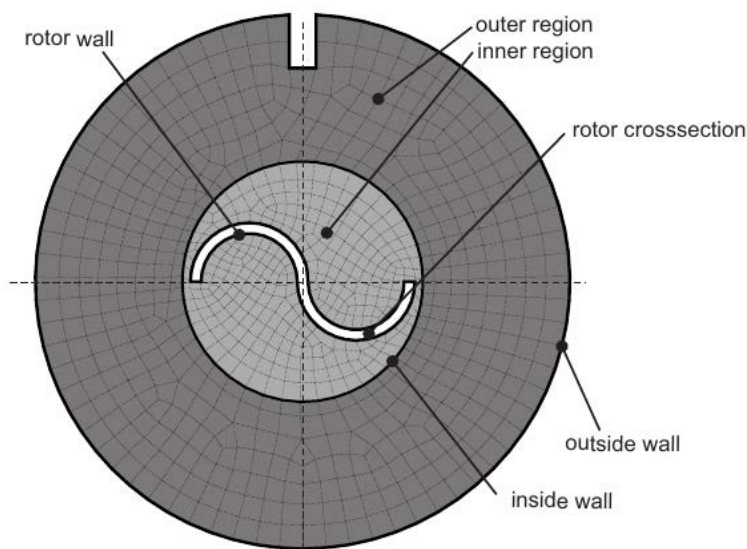


Abbildung 6: Netz des Savonius-Rotors.. Genommen aus (Berger, 2014)

Mit dem erstellten Netz besteht der nächste Schritt darin, eine Case-Datei anzulegen, die die Strömungscharakteristiken sowie die Informationen über den rotierenden Bezugskörper enthält. Um einen rotierenden Bezugskörper zu erhalten, wird die Zellzonenbedingung der inneren Region auf „Frame Motion“ gesetzt, wobei die Rotationsgeschwindigkeit in rad/s als Eingangsparameter festgelegt wird. Anschließend wird die Mischwand als rotierende Wand mit einer Geschwindigkeit von 0 rad/s definiert, da die Wand in diesem Modus zusammen mit dem zuvor erstellten Bezugskörper rotiert. Auf diese Weise ist der rotierende Savonius-Rotor korrekt abgebildet. Als Ausgangsparameter wird das Rotors das wirkende Drehmoment erfasst.

Um die Bewegungsgleichung nun zu lösen, wird sie im Simulink-Umfeld modelliert (Abbildung 7).

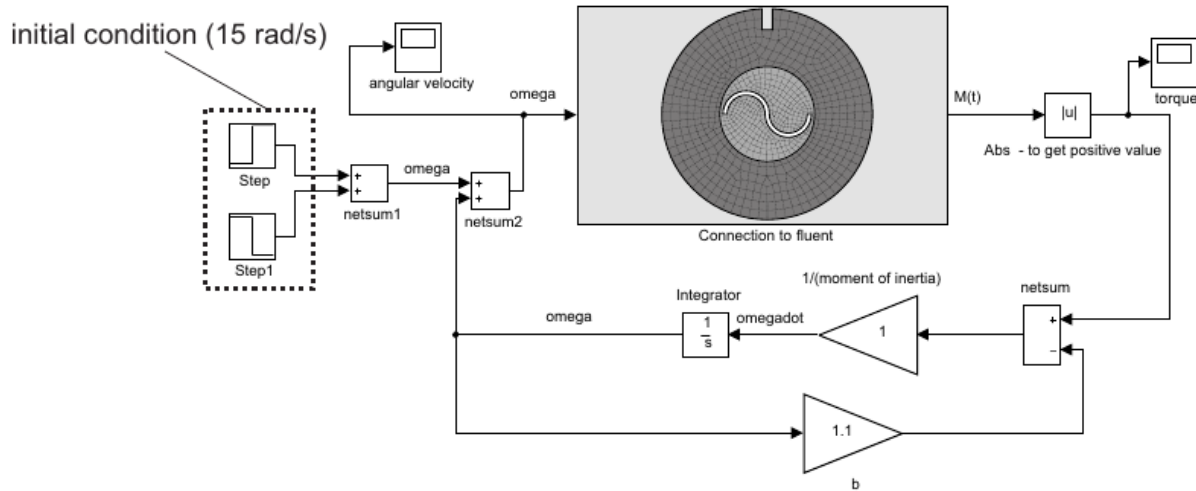


Abbildung 7: Simulink Modell des Savonius-Rotors.. Genommen aus (Berger, 2014)

Die Anfangsbedingung wird auf das Modell angewendet. Bei Simulationszeit 0 Sekunden wird eine fallende Flanke erzeugt, um die Anfangsbedingung zu realisieren. Das auf den Savonius-Rotor wirkende Moment wird vom FLUENT-Solver berechnet. Abbildung 8 zeigt das Ergebnis der Simulation. Aufgrund der Anfangsbedingung startet der Winkelgeschwindigkeitsverlauf bei 15 rad/s. Die Parameter für M und I werden so gewählt, dass die Lösung der Differentialgleichung eine exponentielle Funktion ergibt.

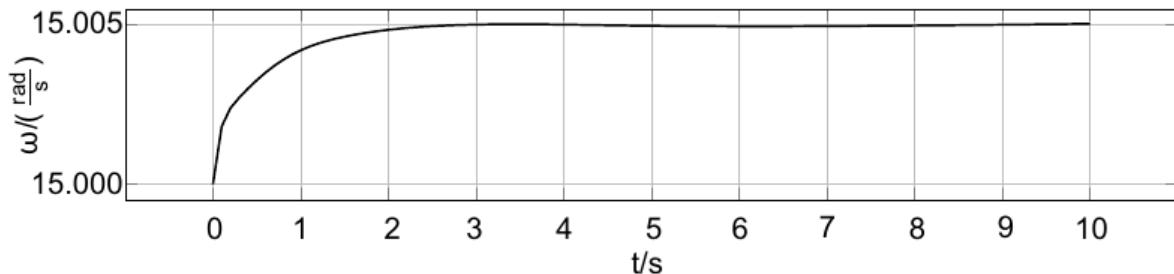


Abbildung 8: Ergebnis des Savonius-Rotors.. Genommen aus (Berger, 2014)

4. SCHLUSSFOLGERUNG

Die Kopplung der Programme erweitert die Möglichkeiten, komplexere Probleme in kürzerer Zeit zu lösen. Der Trend geht dahin, dass niemand mehr daran interessiert ist, Lösungen aus farbigen Bildern oder Filmen abzuleiten. Die grafische Programmieroberfläche von Simulink erleichtert und erweitert die Nutzung von FLUENT erheblich. Das neu entwickelte Plug-in bietet einen einfachen Weg, mit FLUENT zu kommunizieren. Der einzige Nachteil des Werkzeugs besteht darin, dass Lizenzen für FLUENT und MATLAB/Simulink erforderlich sind, da bislang kein vergleichbares, frei verfügbares Open-Source-Programm existiert.

Literaturverzeichnis

Berger, M. (2014). *Closed loop control of a virtual plate wheel*. MCI Innsbruck.

Munteanu, S. (2014). *How to make matlab apps for fluent*. <http://www.ansys-blog.com/how-to-make-matlab%02apps-for-fluent/>